Práctico 2 : Git y GitHub

Objetivo

El estudiante desarrollará competencias para trabajar con Git y GitHub, aplicando conceptos fundamentales de control de versiones, colaboración en proyectos y resolución de conflictos, en un entorno simulado y guiado.

Resultados de aprendizaje:

1. Comprender los conceptos básicos de Git y GitHub: Identificar y explicar los principales términos y procesos asociados con Git y GitHub, como repositorios, ramas, commits, forks, etiquetas y repositorios remotos.
2. Manejar comandos esenciales de Git: Ejecutar comandos básicos para crear, modificar, fusionar y gestionar ramas, commits y repositorios, tanto en local como en remoto.
3. Aplicar técnicas de colaboración en GitHub: Configurar y utilizar repositorios remotos, realizar forks, y gestionar pull requests para facilitar el trabajo colaborativo.
4. Resolver conflictos en un entorno de control de versiones: Identificar, analizar y solucionar conflictos de merge generados en un flujo de trabajo con múltiples ramas.

Actividades:

1. Contestar las siguientes preguntas utilizando las guías y documentación proporcionada (Desarrollar las respuestas):
    - ¿Qué es GitHub?

      > Es una plataforma de desarrollo colaborativo basada en la nube que permite gestionar repositorios Git, facilitando el control de versiones y la colaboración en proyectos de software.

    - ¿Cómo crear un repositorio en GitHub?

      > Iniciar sesión en GitHub. Hacer clic en el botón "+" y seleccionar "New repository". Ingresar un nombre, una descripción y seleccionar la visibilidad (público o privado). Hacer clic en "Create repository".

    - ¿Cómo crear una rama en Git?

      > git branch branch-name

    - ¿Cómo cambiar a una rama en Git?

      > git checkout branch-name O en versiones recientes de git: git switch branch-name

    - ¿Cómo fusionar ramas en Git?

      > Cambiar a la rama donde se desea fusionar, por ejemplo main: git checkout main Fusionar la rama branch-name con main: git merge branch-name

    - ¿Cómo crear un commit en Git?

      > Agregar los cambios al área de preparación: git add . Ejecutar el commit: git commit -m "Mensaje del commit"

    - ¿Cómo enviar un commit a GitHub?

      > git push origin branch-name

    - ¿Qué es un repositorio remoto?

      > Es una versión de un repositorio almacenada en un servidor, como GitHub, que permite la colaboración entre desarrolladores.

    - ¿Cómo agregar un repositorio remoto a Git?

      > git remote add origin URL-del-repositorio

    - ¿Cómo empujar cambios a un repositorio remoto?

      > git push origin branch-name

    - ¿Cómo tirar de cambios de un repositorio remoto?

      > git pull origin branch-name

    - ¿Qué es un fork de repositorio?

      > Es una copia de un repositorio que permite hacer cambios sin afectar al original. Es útil para contribuciones en proyectos de código abierto.

    - ¿Cómo crear un fork de un repositorio?

      > Ir al repositorio en GitHub. Hacer clic en el botón "Fork" en la parte superior derecha.

    - ¿Cómo enviar una solicitud de extracción (pull request) a un repositorio?

      > Subir los cambios a tu fork. Ir a la página del repositorio original. Hacer clic en "Pull Requests" y luego en "New Pull Request". Seleccionar las ramas y envíar la solicitud.

    - ¿Cómo aceptar una solicitud de extracción?

      > Ir a la sección "Pull Requests" del repositorio. Abrir la solicitud y revisar los cambios. Si está todo correcto, hacer clic en "Merge Pull Request".

    - ¿Qué es un etiqueta en Git?

      > Es una referencia a un punto específico en el historial, útil para marcar versiones.

    - ¿Cómo crear una etiqueta en Git?

> git tag -a v1.0 -m "Version 1.0"

- ¿Cómo enviar una etiqueta a GitHub?

  > git push origin v1.0

- ¿Qué es un historial de Git?

  > Es el registro de todos los commits realizados en un repositorio.

- ¿Cómo ver el historial de Git?

  > git log

- ¿Cómo buscar en el historial de Git?

  > git log --grep="texto"

- ¿Cómo borrar el historial de Git?

  > No se recomienda, pero si es necesario se utiliza el siguiente comando, donde "n" es la cantidad de commits a modificar: git rebase -i HEAD~n

- ¿Qué es un repositorio privado en GitHub?

  > Es un repositorio en GitHub que solo puede ser accedido por los usuarios autorizados.

- ¿Cómo crear un repositorio privado en GitHub?

  > Al crear un repositorio, selecciona "Private" en las opciones de visibilidad.

- ¿Cómo invitar a alguien a un repositorio privado en GitHub?

  > Ir a "Settings" del repositorio. En "Collaborators", agregar la persona con su nombre de usuario o correo.

- ¿Qué es un repositorio público en GitHub?

  > Un repositorio que es visible para cualquier usuario de GitHub.

- ¿Cómo crear un repositorio público en GitHub?

  > Al crear un repositorio, seleccionar "Public" en las opciones de visibilidad.

- ¿Cómo compartir un repositorio público en GitHub?

  > A través de la URL del repositorio público.

2. Realizar la siguiente actividad:

   - Crear un repositorio https://github.com/m415x/UTN-TUPaD-P1-TP2-A2

     - Dale un nombre al repositorio.
     - Elije el repositorio sea público.
     - Inicializa el repositorio con un archivo.

   - Agregando un Archivo

     - Crea un archivo simple, por ejemplo, "mi-archivo.txt".
     - Realiza los comandos git add. y git commit -m "Agregando mi-archivo.txt" en la línea de comandos.
     - Sube los cambios al repositorio en GitHub con git push origin main (o el nombre de la rama correspondiente).

   - Creando Branchs

     - Crear una Branch
     - Realizar cambios o agregar un archivo
     - Subir la Branch

3. Realizar la siguiente actividad:

   - Paso 1: Crear un repositorio en GitHub https://github.com/m415x/UTN-TUPaD-P1-TP2-A3-conflict-exercise

     - Ve a GitHub e inicia sesión en tu cuenta.
     - Haz clic en el botón "New" o "Create repository" para crear un nuevo repositorio.
     - Asigna un nombre al repositorio, por ejemplo, conflict-exercise.
     - Opcionalmente, añade una descripción.
     - Marca la opción "Initialize this repository with a README".
     - Haz clic en "Create repository".

   - Paso 2: Clonar el repositorio a tu máquina local

     - Copia la URL del repositorio https://github.com/m415x/UTN-TUPaD-P1-TP2-A3-conflict-exercise.
     - Abre la terminal o línea de comandos en tu máquina.
     - Clona el repositorio usando el comando:

       ```
       git clone https://github.com/m415x/UTN-TUPaD-P1-TP2-A3-conflict-exercise
       ```

     - Entra en el directorio del repositorio:

       ```
       cd UTN-TUPaD-P1-TP2-A3-conflict-exercise
       ```

   - Paso 3: Crear una nueva rama y editar un archivo

- Crea una nueva rama llamada feature-branch:

```
git checkout -b feature-branch
```

- Abre el archivo `README.md` en un editor de texto y añade una línea nueva, por ejemplo: "Este es un cambio en la feature branch."
- Guarda los cambios y haz un commit:

```
git add README.md
git commit -m "Added a line in feature-branch"
```

- Paso 4: Volver a la rama principal y editar el mismo archivo
  - Cambia de vuelta a la rama principal (main):

```
git checkout main
```

  - Edita el archivo `README.md` de nuevo, añadiendo una línea diferente: "Este es un cambio en la main branch."
  - Guarda los cambios y haz un commit:

```
git add README.md
git commit -m "Added a line in main branch"
```

- Paso 5: Hacer un merge y generar un conflicto
  - Intenta hacer un merge de la feature-branch en la rama main:

```
git merge feature-branch
```

  - Se generará un conflicto porque ambos cambios afectan la misma línea del archivo `README.md`.
- Paso 6: Resolver el conflicto
  - Abre el archivo `README.md` en tu editor de texto. Verás algo similar a esto:

```
<<<<<<< HEAD
Este es un cambio en la main branch.
=======
Este es un cambio en la feature branch.
> > > > > > > feature-branch
```

  - Decide cómo resolver el conflicto. Puedes mantener ambos cambios, elegir uno de ellos, o fusionar los contenidos de alguna manera.
  - Edita el archivo para resolver el conflicto y guarda los cambios (Se debe borrar la parte del texto que no se quiera dejar).
  - Añade el archivo resuelto y completa el merge:

```
git add README.md
git commit -m "Resolved merge conflict"
```

- Paso 7: Subir los cambios a GitHub
  - Sube los cambios de la rama main al repositorio remoto en GitHub:

```
git push origin main
```

  - También sube la feature-branch si deseas:

```
git push origin feature-branch
```

- Paso 8: Verificar en GitHub
  - Ve a tu repositorio en GitHub y revisa el archivo `README.md` para confirmar que los cambios se han subido correctamente.
  - Puedes revisar el historial de commits para ver el conflicto y su resolución.